Globus Toolkit® v2.2 (GT2) Tutorial

Data Management

The Globus Project™

Argonne National Laboratory

USC Information Sciences Institute

http://www.globus.org/

Sources of Information

- http://www.globus.org/toolkit/documentation/
- discuss@globus.org
- globus_ftp_control.h
- http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf

Data Management Services

- Data transfer and access
  - **GASS:** Simple, multi-protocol file transfer tools; integrated with GRAM
  - **GridFTP:** Provides high-performance, reliable data transfer for modern WANs
- Data replication and management
  - **Replica Catalog:** Provides a catalog service for keeping track of replicated datasets
  - **Replica Management:** Provides services for creating and managing replicated datasets

the globus toolkit™
www.globustoolkit.org

GASS
Remote I/O and Staging

- Used by GRAM to:
  - Pull executable from remote location
  - Move stdin/stdout/stderr from/to a remote location
- Access files from a remote location

What is GASS?
Global Access to Secondary Storage

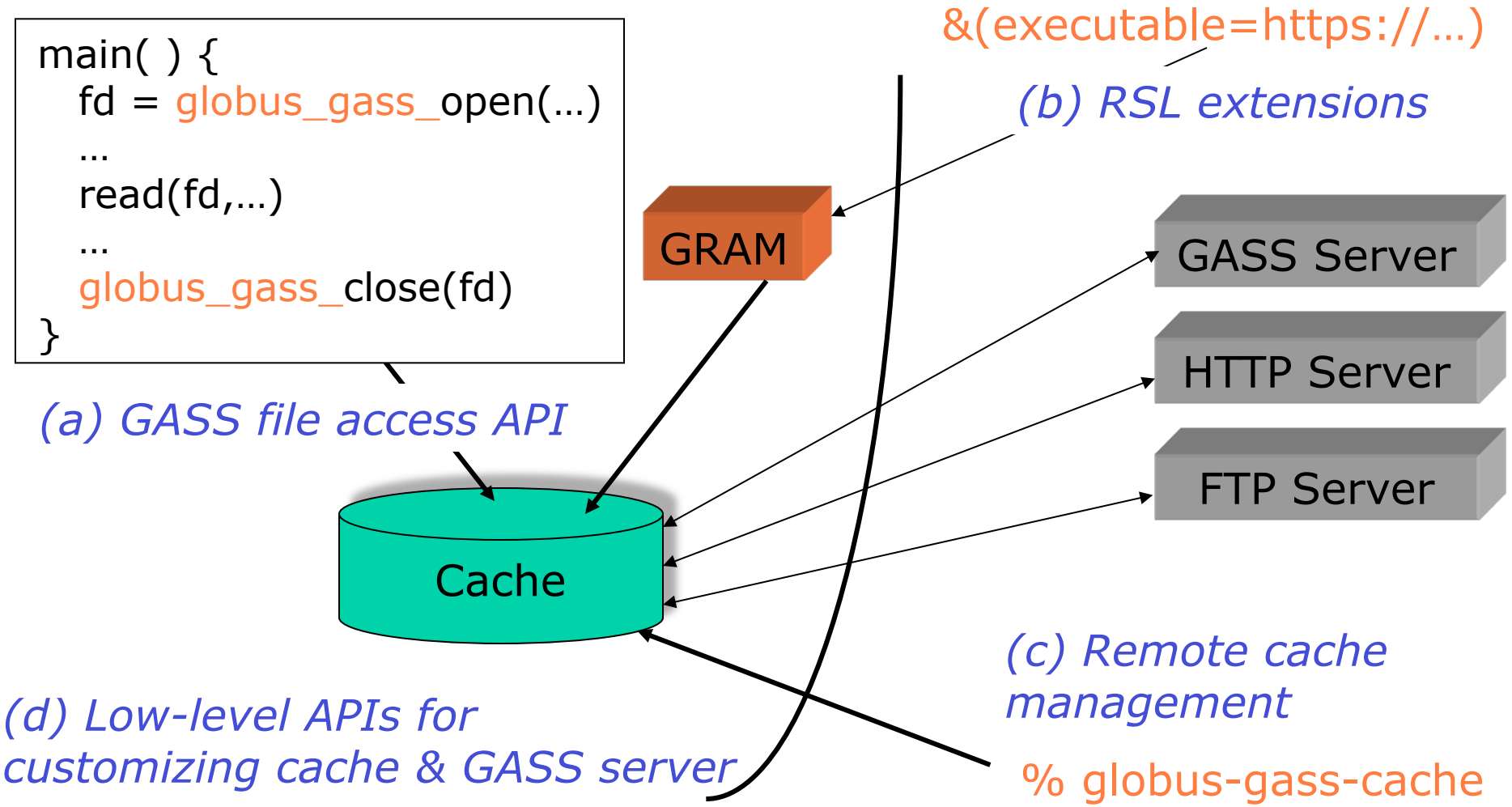
(a) GASS file access API
- Replace open/close with globus_gass_open/close; read/write calls can then proceed directly

(b) RSL extensions
- URLs used to name executables, stdout, stderr

(c) Remote cache management utility

(d) Low-level APIs for specialized behaviors

GASS Architecture



GASS File Naming

- URL encoding of resource names

  https://quad.mcs.anl.gov:9991/~bester/myjob

  protocol server address file name

- Other examples

  https://pitcairn.mcs.anl.gov/tmp/input_dataset.1

  https://pitcairn.mcs.anl.gov:2222/./output_data

  http://www.globus.org/~bester/input_dataset.2

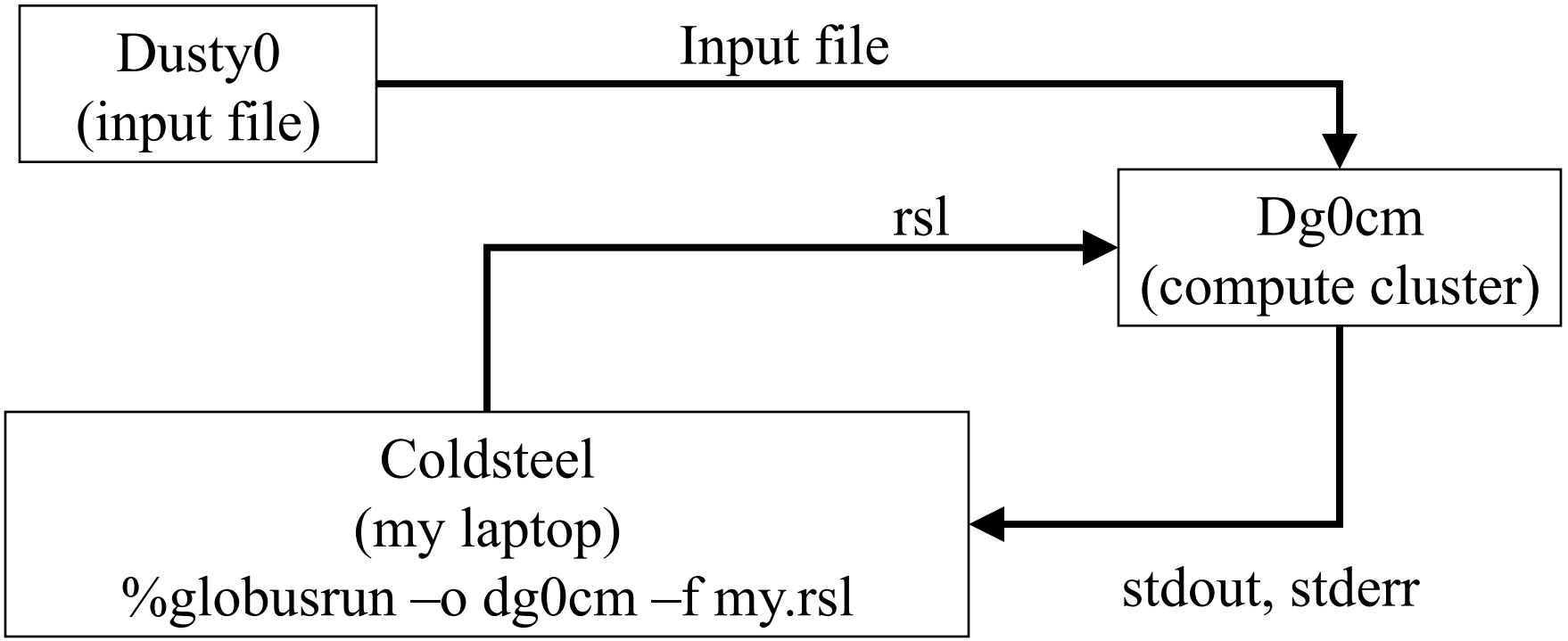
- Currently supports http, https, ftp, and gsiftp

GASS RSL Extensions

- executable, stdin, stdout, stderr can be local files or URLs
- executable and stdin loaded into local cache before job begins (on front-end node)
- stdout, stderr handled via GASS append mode
- Many more new ones in GRAM 1.6 that will be covered in that section.
- Cache cleaned after job completes

GASS/RSL Example

```
&(executable=/home/allcock/myexe)
  (stdin=gsiftp://dusty0/home/allcock/myin)
  (stdout=https://coldsteel:1234/home/allcock/output)
  (stderr=https://coldsteel:1234/dev/stdout)
```



Example GASS Applications

- On-demand, transparent loading of data sets
- Caching of (small) data sets
- Automatic staging of code and data to remote supercomputers
  - GridFTP better suited to staging of large data sets
  - GASS can use GridFTP, but can't set parameters like buffer size, and parallelism
- (Near) real-time logging of application output to remote server

GASS Examples

- globus-job-run pitcairn –s myscript.sh

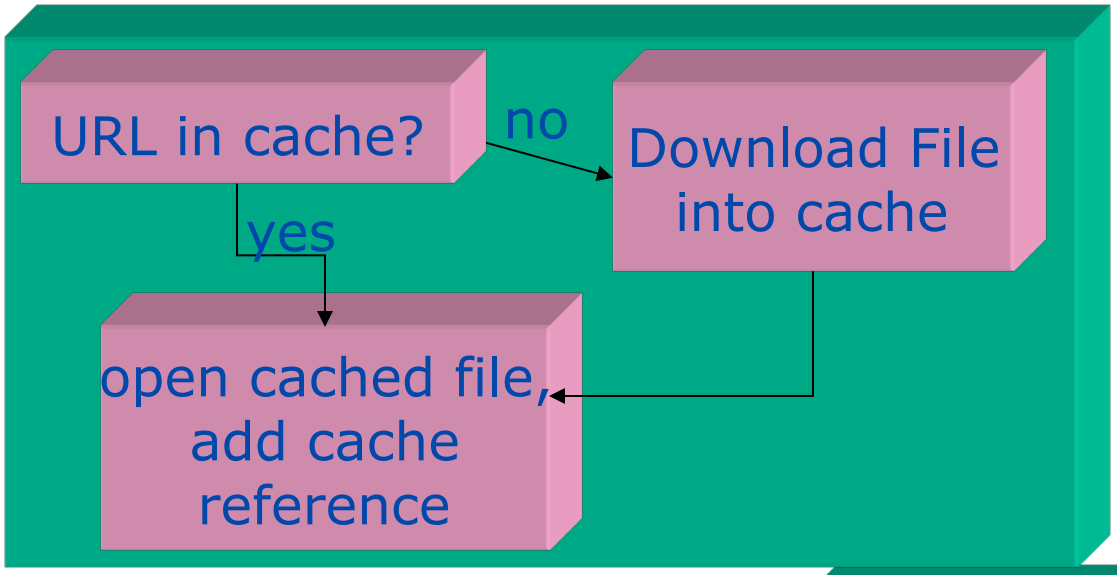
GASS File Access API

- Minimum changes to application
- globus_gass_open(), globus_gass_close()
  - Same as open(), close() but use URLs instead of filenames
  - Caches URL in case of multiple opens
  - Return descriptors to files in local cache or sockets to remote server
- globus_gass_fopen(), globus_gass_fclose()

GASS File Access API (cont)

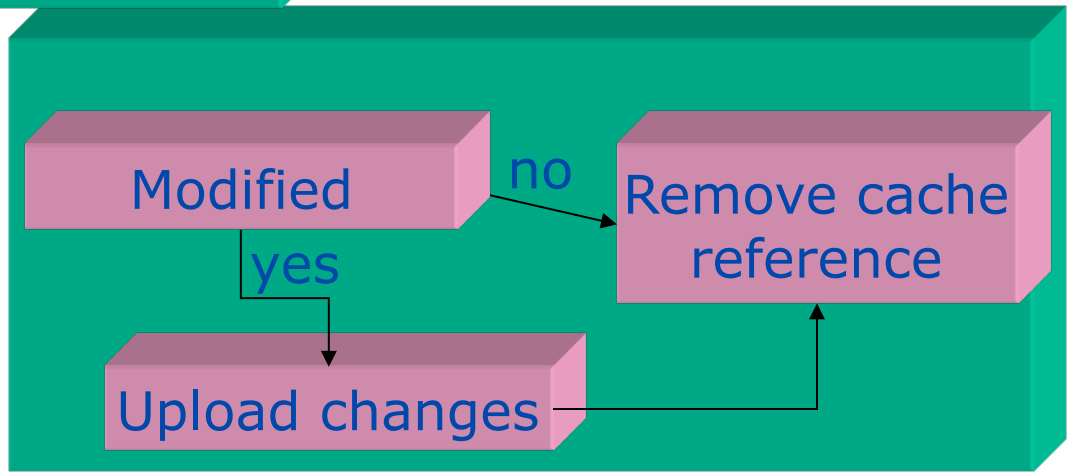
- Support for different access patterns
  - Read-only (from local cache)
  - Write-only (to local cache)
  - Read-write (to/from local cache)
  - Write-only, append (to remote server)

globus_gass_open()/close()



globus_gass_open()

globus_gass_close()



GASS File API Semantics

- Copy-on-open to cache if not truncate or write-only append *and* not already in cache
- Copy on close from cache if not read only *and* not other copies open
- Multiple globus_gass_open() calls share local copy of file
- Append to remote file if write only append: e.g., for stdout and stderr
- Reference counting keeps track of open files

Remote Cache Management Utilities

- Remote management of caches, for
  - Prestaging/poststaging of files
  - Cache cleanup and management
- Support operations on local & remote caches
- Functionality encapsulated in a program: globus-gass-cache

GASS Cache Semantics

- For each "file" in the cache, we record
  - Local file name
  - URL (i.e., the remote location)
  - Reference count: a set of tagged references
- Tags associated with references allow clean up of cache, e.g. following failure
  - Tag is job_manager_contact (if file accessed via file access API) or programmer-specified
  - Commands allow "remove all refs with tag T"

globus-gass-cache Specification

globus-gass-cache op [-r resource] [-t tag] URL

- Where op is one of
  - add : add URL to cache with tag
  - delete : remove one reference of tag for URL
  - cleanup_tag : remove all refs of tag for URL
  - cleanup_url : remove specified URL from cache
  - list : list contents of cache
- URL is optional for cleanup_tag and list
- If resource not specified, default to local cache

globus-gass-cache Examples

globus-gass-cache add -t experiment1 https://host:port/file

- Add file "file" (located at https://host:port) to the local cache; label reference with tag "experiment1"

globus-gass-cache add -r tuva.mcs.anl.gov-fork \
https://host:port/file

- Add file "file" (located at x-gass://host:port) to the cache at tuva.mcs.anl.gov-fork

globus_gass_cache

- Module for manipulating the GASS cache
  - globus_gass_cache_open(), …_close()
  - globus_gass_cache_add(), …_add_done()
  - globus_gass_cache_delete_start(), …_delete()
  - globus_gass_cache_cleanup_tag()
  - globus_gass_cache_cleanup_file()
  - globus_gass_cache_list()
- This modules does NOT fill in the contents of the cache files. It just handles manages naming and lifetimes of files.

globus_gass_transfer

- Common API for transferring remote files/ data over various protocols
  - http and https currently supported
  - ftp will be supported in future release
- Supports put and get operations on an URL
- Allows for efficient transfer to/from files or direct to/from memory
- Allows any application to easily add customized file/data service capabilities

globus_gass_copy

- Simple API for copying data from a source to a destination
  - URL used for source and destination
  - http(s), (gsi)ftp, file
  - When transferring from ftp to ftp, it uses 3rd party transfer (I.e. client mediated, direct server-to-server transfer)
- globus-url-copy program is simple wrapper around the globus_gass_copy API

globus-gass-server

- Simple file server
  - Run by user wherever necessary
  - Secure https protocol, using GSI
  - APIs for embedding server into other programs
- Example

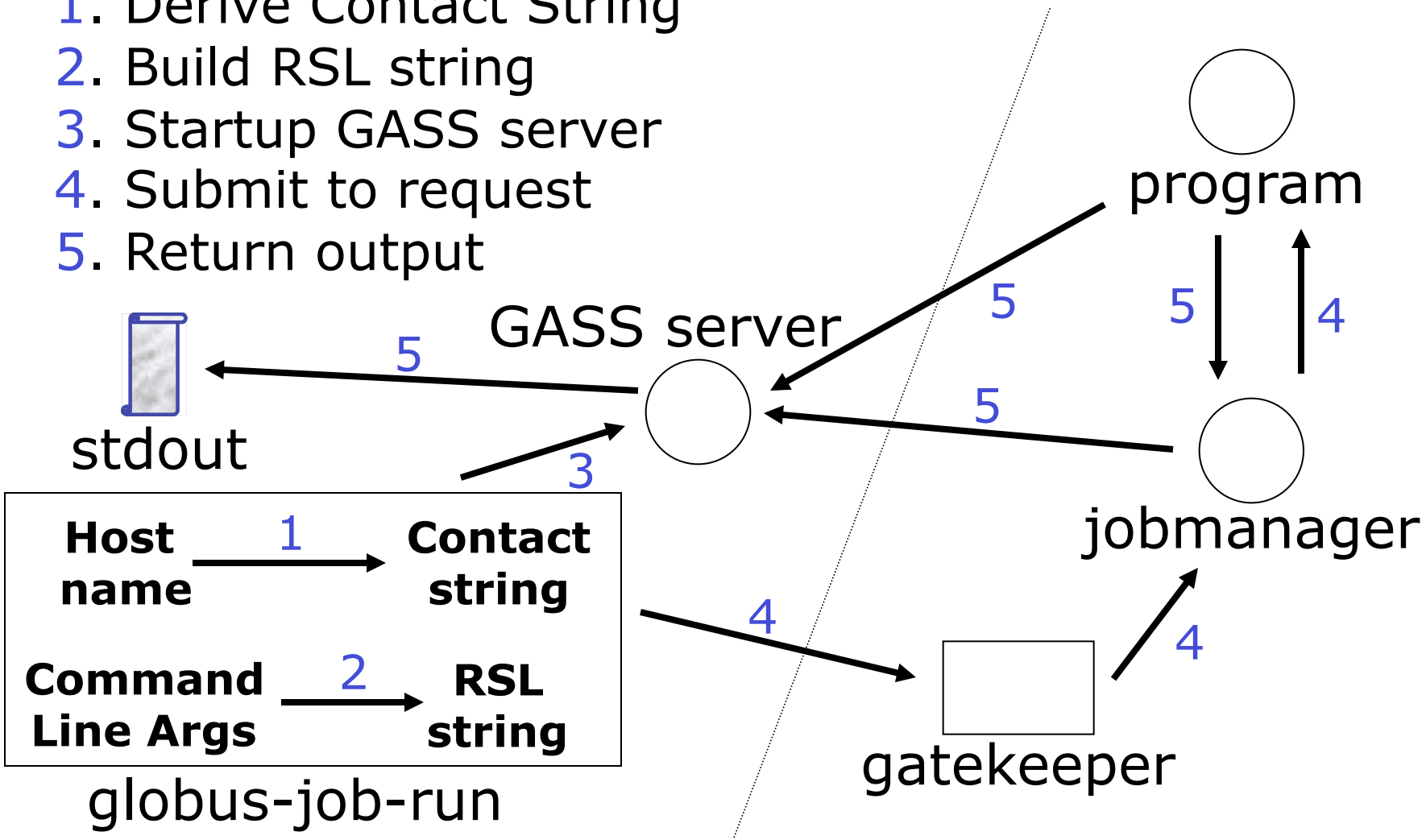
  globus-gass-server –r –w -t
  - -r: Allow files to be read from this server
  - -w: Allow files to be written to this server
  - -t: Tilde expand (~/… → $(HOME)/…)
  - -help: For list of all options

globus_gass_server_ez

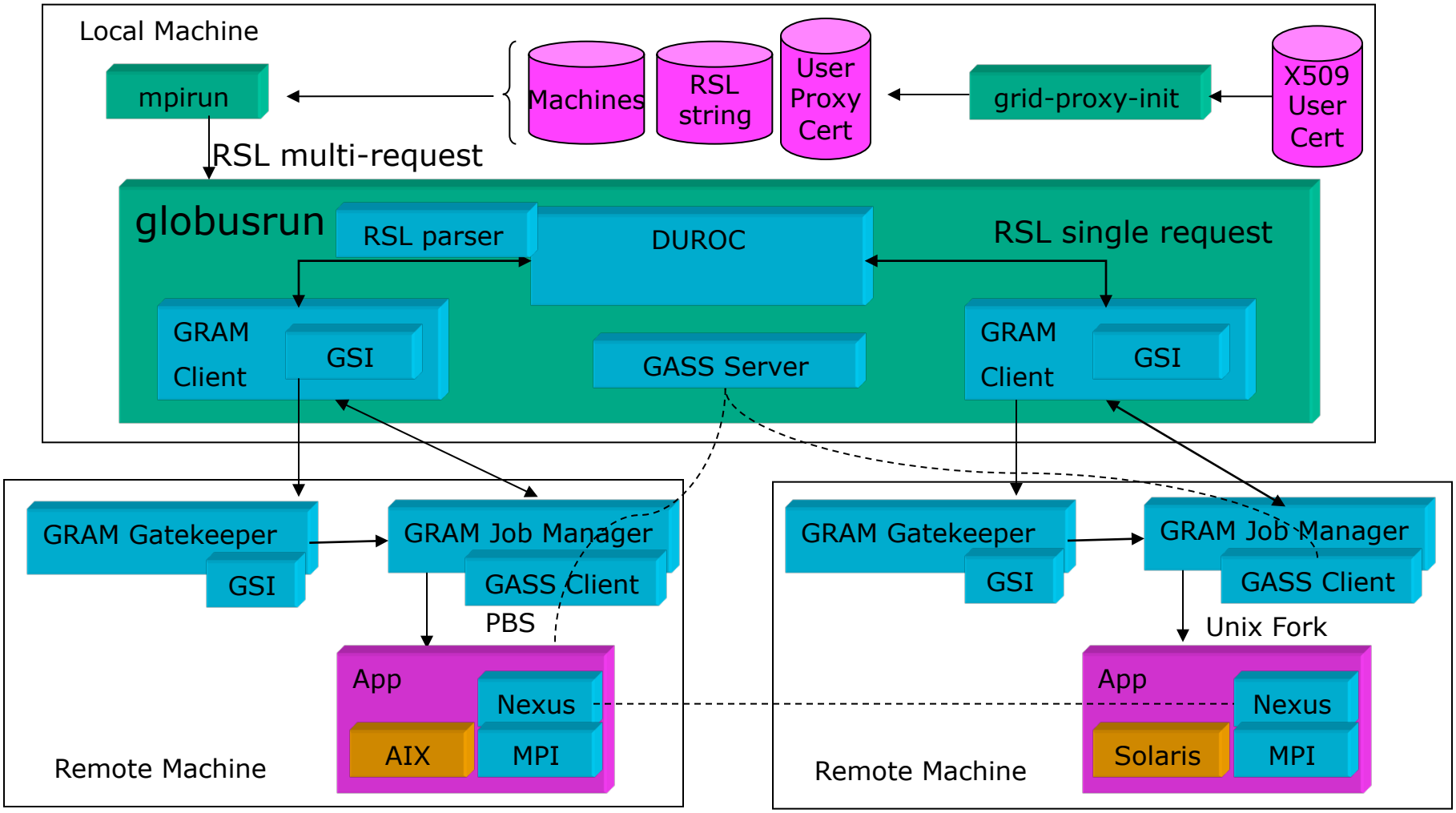
- Very simply API for adding file service to any application
  - Wrapper around globus_gass_transfer
- globusrun uses this module to support executable staging, stdout/err redirection, and remote file access

GRAM & GASS: Putting It Together

1. Derive Contact String
2. Build RSL string
3. Startup GASS server
4. Submit to request
5. Return output



Globus Components In Action



But GASS Is Not Enough

- GASS is designed for convenient access to smaller files
  - Integrated with GRAM
  - Simple remote access APIs
- What about high-end data transfer and access needs?
  - High-performance transfer
  - Third party (client mediated) transfer
  - Richer access patterns
- GridFTP is the answer…

GridFTP: Basic Approach

- FTP protocol is defined by several IETF RFCs
- Start with most commonly used subset
  - Standard FTP: get/put etc., 3rd-party transfer
- Implement standard but often unused features
  - GSS binding, extended directory listing, simple restart
- Extend in various ways, while preserving interoperability with existing servers
  - Striped/parallel data channels, partial file, automatic & manual TCP buffer setting, progress monitoring, extended restart

GridFTP APIs (in v2)

- globus_ftp_control
  - Provides access to low-level GridFTP control and data channel operations.
- globus_ftp_client
  - Provides typical GridFTP client operations.
- globus_gass_copy
  - Provides the ability to easily start and manage multiple data transfers using GridFTP, HTTP, local file, and memory operations

globus-url-copy

- This is the GridFTP client tool provided with the Globus Toolkit V2.2™
- It takes a source URL and destination URL and will do protocol conversion for http, https, FTP, gsiftp, and file (file must be local).
- globus-url-copy program is simple wrapper around globus-gass-copy

A few Caveats

- This is a client server model. I.e., there must be a server running somewhere.
- Today, that means one of our GT2 wuftpd based servers.
- We hope other people are going to write servers.
- We have an embeddable server on our list of things to do, but don't know when we will have the resources.
- Implementing the protocol is not trivial

globus_ftp_control

- Low level GridFTP driver
  - Control channel management
    - > Both client and server sides
    - > Handles message framing, security, etc
  - Data channel management
    - > Symmetric for client and server sides
    - > Designed for performance: caller controls buffer management, no data copies needed
- Must understand details of GridFTP protocol to use this API
  - Intended for custom GridFTP client and server developers

Nomenclature

- Gsi-wuftpd and gsi-ncftp are NOT GridFTP clients or servers
- The GT2 GridFTP is based on the wuftpd and IS a GridFTP server.
- To be a GridFTP server, you must be compliant with the Protocol Doc
- Ours is the only one right now, but hopefully there will be other implementations in the future.

The GridFTP Protocol

- Based on 4 RFC's and our extensions
- RFC 959: The base FTP protocol document
- RFC 2228: Security Extensions
- RFC 2389: Feature Negotioation and support for command options
- IETF Draft: Stream Mode restarts, standard file listings

The GridFTP Protocol (Cont)

- Our Extensions:
  - SPOR/SPAS
  - ERET/ESTO
  - SBUF/ABUF
  - DCAU
  - Mode E
  - Options to RETR
  - FEAT

The GridFTP Protocol (Cont)

- Command Response Protocol
- Issue a command, get only responses to that command until it is completed, then you can issue another command

The GridFTP Protocol (Cont)

- Simple Commands have Complex Protocol Exchanges.
- PUT (transfer file from my client to server)
  - TYPE I, MODE E, PASV, STOR <FileName>
  - Client can form multiple connections
  - Restart and Performance Markers over the control channel
  - Specific replies and defined state machine.

globus_ftp_client

- Functionality
  - get, put, third_party_transfer
    - > Variants: normal, partial file, extended
  - delete, mkdir, rmdir, move
    - > Note no "cd". All operations use URLs with full paths
  - list, verbose_list
  - modification_time, size, exists
  - Hides the state machine
  - PlugIn Architecture provides access to interesting events.
- All data transfer is to/from memory buffers
  - Facilitates wide range of clients

Writing a GridFTP Client

- Module Activation / Initialization
- Check Features
- Select Mode
- Set Attributes
- Enable any needed plug-ins
- Execute the operation
- Module Deactivation / Clean up

Initialization

- globus_module_activate(GLOBUS_FTP_CLIENT_MODULE)
- Must be called in any program that use the client library.
- Will automatically call module_activate for any required lower level modules (like globus_io)

Checking Features

- Right now, you would have to go to the control library and send the FEAT command yourself.
- We are adding a function that will allow you to check to see if a feature is supported.

Attributes

- Very powerful feature and control much of the functionality
- Two types of attributes:
  - Handle Attributes: Good for an entire session and independent of any specific Operation
  - Operation Attributes: Good for a single operation.
- Files:
  - globus_ftp_client_attr.c
  - globus_i_ftp_client.h

Attributes (Cont)

- Handle Attributes:
  - Initialize/Destroy/Copy Attribute Handle
  - Connection Caching: Either all, or URL by URL.
  - Plugin Management: Add/Remove Plugins

Attributes (Cont)

- Operation Attributes
  - Parallelism
  - Striped Data Movement
  - Striped File Layout
  - TCP Buffer Control
  - File Type
  - Transfer Mode
  - Authorization/Privacy/Protection
- Functions
  - globus_ftp_client_operationattr_set_<attribute>(&attr, &<attribute_struct>)
  - globus_ftp_client_operationattr_get_<attribute>(&attr, &<attribute_struct>)

Attributes (Cont)

- Example Code (structs and enums in globus_ftp_control.h):

```
globus_ftp_client_handle_t              handle;
globus_ftp_client_operationattr_t       attr;
globus_ftp_client_handleattr_t          handle_attr;
globus_size_t                           parallelism_level = 4;
globus_ftp_control_parallelism_t        parallelism;
globus_ftp_control_layout_t             layout;

globus_module_activate(GLOBUS_FTP_CLIENT_MODULE);
globus_ftp_client_handleattr_init(&handle_attr);
globus_ftp_client_operationattr_init(&attr);
parallelism.mode = GLOBUS_FTP_CONTROL_PARALLELISM_FIXED;
parallelism.fixed.size = parallelism_level;
globus_ftp_client_operationattr_set_mode(&attr,
    GLOBUS_FTP_CONTROL_MODE_EXTENDED_BLOCK);
globus_ftp_client_operationattr_set_parallelism(&attr, &parallelism);
globus_ftp_client_handle_init(&handle,  &handle_attr);
```

Mode S versus Mode E

- Mode S is stream mode as defined by RFC 959.
  - No advanced features accept simple restart
- Mode E enables advanced functionality
  - Adds 64 bit offset and length fields to the header.
  - This allows discontiguous, out-of-order transmission and along with the SPAS and SPOR commands, enable parallelism and striping.
- Command:

```
globus_ftp_client_operationattr_set_mode(&attr, GLOBUS_FTP_CONTROL_MODE_EXTENDED_BLOCK);
```

Plug-Ins

- Interface to one or more plug-ins:
  - Callouts for all interesting protocol events
    - > Allows monitoring of performance and failure
  - Callins to restart a transfer
    - > Can build custom restart logic
- Included plug-ins:
  - Debug: Writes event log
  - Restart: Parameterized automatic restart
    - > Retry N times, with a certain delay between each try
    - > Give up after some amount of time
  - Performance: Real time performance data

Plug-Ins (Cont.)

- A plugin is created by defining a globus_ftp_client_plugin_t which contains the function pointers and plugin-specific data needed for the plugin's operation. It is recommended that a plugin define a a globus_module_descriptor_t and plugin initialization functions, to ensure that the plugin is properly initialized.
- Every plugin must define **copy** and **destroy** functions. The copy function is called when the plugin is added to an attribute set or a handle is initialized with an attribute set containing the plugin. The destroy function is called when the handle or attribute set is destroyed.

Plug-Ins (Cont.)

- Essentially filling in a structure of function pointers:
  - Operations (Put, Get, Mkdir, etc)
  - Events (command, response, fault, etc)
- Called only if both the operation and event have functions defined
- Filtered based on command_mask

Plug-Ins (Cont.)

- Coding:
  - globus_ftp_client_plugin_t *plugin;
  - globus_ftp_client_plugin_set_<type>_func
    - > Macro to make loading the struct easier
  - globus_ftp_client_handleattr_add_plugin(attr, plugin)
- Files:
  - globus_ftp_client_plugin.h
  - globus_ftp_client.h
  - globus_ftp_client_plugin.c
  - Also some internal .h files

the globus toolkit™
www.globustoolkit.org

High Level Calls

- globus_ftp_client_put/get/3rd Party
- Function signature:

globus_result_t **globus_ftp_client_get**
(**globus_ftp_client_handle_t** *handle,
const char *url,
globus_ftp_client_operationattr_t *attr,
globus_ftp_client_restart_marker_t *restart,
globus_ftp_client_complete_callback_t
complete_callback,
void *callback_arg)

Example: globus_ftp_client_put_test.c

Parallel Put/Get

- Parallelism is hidden. You are not required to do anything other than set the attributes, though you may want to for performance reasons.
- Doc needs to be updated. Does not have enums or structures. Look in globus_ftp_control.h

Making GridFTP Go… FAST!

- The Chain is only as strong as the weakest Link
- OS Limitations on Streams and buffers
  - Buffer size limits (defaults, Max)
  - /etc/sysctl.conf (Linux)
  - We use 64K default, 8MB Max per socket
  - # of sockets per process and total
- Note that with striping and parallelism you can end up with a lot of memory and streams in a real hurry.

Making GridFTP Go… FAST!

- NIC's: Gigabit or go Slow
  - Can't really recommend a brand, because it is so system dependant.
  - Our experience: SysKonnect 98 series are good, NetGear GA620 are good, not much experience with GA621, SysKonnect 9D supposed to be fast, less expensive, but higher latency (not relevant for GridFTP)
  - Check your configuration
    - > Auto Duplex selection rarely works
    - > Interrupt Coalescing
    - > HW Checksumming

Making GridFTP Go… FAST! (Cont)

- Bus: 66MHz (for Intel)
  - We are moving a lot of data: On/Off Disk, In/Out the NIC.
  - We do not use the Linux Zero-Copy stuff. We are looking at it, but it is Linux specific (at least for now)
- CPU: Take Two they are small
  - GigE NIC's take a lot of CPU, so does SW RAID
  - Rumor has it that the PIV Chip Set has low IO rates.
- Disk: Often the biggest Problem
  - IDE limited to between 5 and 20 MB/s
  - Journaling File System is slower, but they are making improvements (ext2 .vs. Reiser .vs. XFS)
  - For Real Speed, use RAID (Software works well if you have enough CPU, otherwise use HW RAID)
  - IDE RAID is now available, but no experience with it

GassCopy API

- globus_result_t **globus_gass_copy_handle_init** (globus_gass_copy_handle_t *handle, globus_gass_copy_handleattr_t *attr)
- globus_result_t **globus_gass_copy_attr_init** (globus_gass_copy_attr_t *attr)
- globus_result_t **globus_gass_copy_attr_set_ftp** (globus_gass_copy_attr_t *attr, globus_ftp_client_operationattr_t *ftp_attr)
- globus_result_t **globus_gass_copy_attr_set_io** (globus_gass_copy_attr_t *attr, globus_io_attr_t *io_attr)

GassCopy API

- globus_result_t **globus_gass_copy_url_to_url** (globus_gass_copy_handle_t *handle, char *source_url, globus_gass_copy_attr_t *source_attr, char *dest_url, globus_gass_copy_attr_t *dest_attr)
- globus_result_t **globus_gass_copy_url_to_handle** (globus_gass_copy_handle_t *handle, char *source_url, globus_gass_copy_attr_t *source_attr, globus_io_handle_t *dest_handle)
- globus_result_t **globus_gass_copy_handle_to_url** (globus_gass_copy_handle_t *handle, globus_io_handle_t *source_handle, char *dest_url, globus_gass_copy_attr_t *dest_attr)

Control Library Overview

- globus_result_t **globus_ftp_control_handle_init** (globus_ftp_control_handle_t *handle)
- globus_result_t **globus_ftp_control_send_command** (globus_ftp_control_handle_t *handle, const char *cmdspec, **globus_ftp_control_response_callback_t** callback, void *callback_arg,...)
- globus_result_t **globus_ftp_control_connect** (globus_ftp_control_handle_t *handle, char *host, unsigned short port, **globus_ftp_control_response_callback_t** callback, void *callback_arg)
- globus_result_t **globus_ftp_control_authenticate** (globus_ftp_control_handle_t *handle, **globus_ftp_control_auth_info_t** *auth_info, globus_bool_t use_auth, **globus_ftp_control_response_callback_t** callback, void *callback_arg)

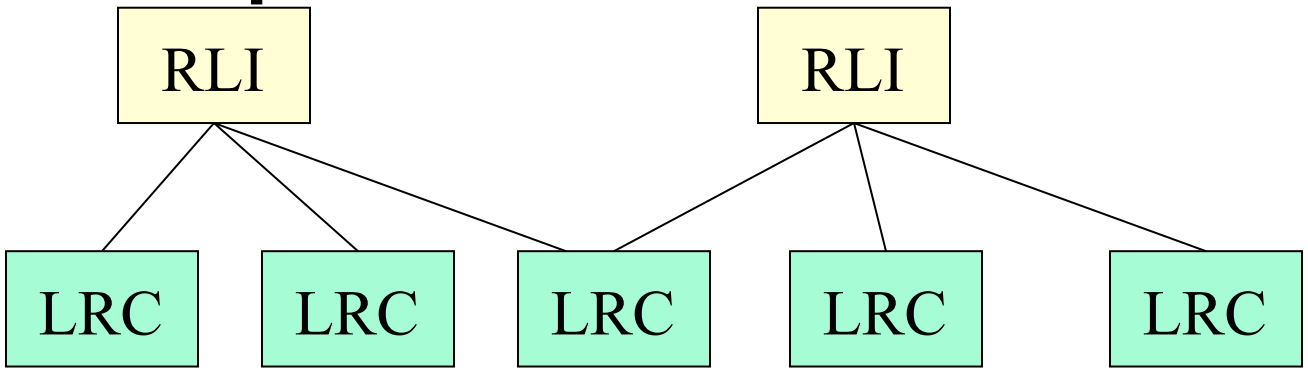
Replica Management Approach

- Replica Catalog API
  - A catalog that represents collections, files, and locations
  - Given a logical filename, at which locations can the file be found?
  - Given a collection, which logical files exist?
- Replica Management API
  - A set of functions for registering files in the replica catalog, publishing files to locations, adding/ removing replicas at other locations
  - Uses Replica Catalog and GridFTP

A Replica Location Service

- **A Replica Location Service (RLS)** is a distributed registry service that records the locations of data copies and allows discovery of replicas
- Maintains mappings between *logical* identifiers and *target names*
  - Physical targets: Map to exact locations of replicated data
  - Logical targets: Map to another layer of logical names, allowing storage systems to move data without informing the RLS
- RLS was designed and implemented in a collaboration between the Globus project and the DataGrid project

Replica Location Indexes



Local Replica Catalogs

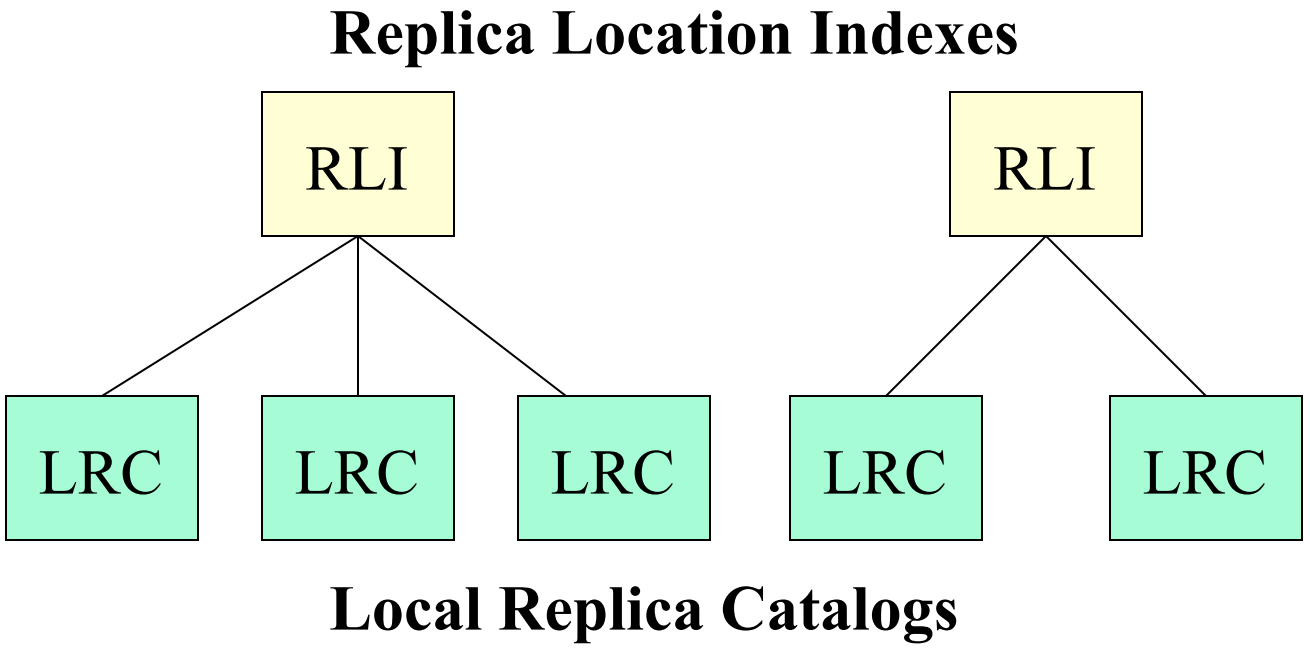
- LRCs contain consistent information about logical-to-target mappings on a site
- RLIs nodes aggregate information about LRCs
- Soft state updates from LRCs to RLIs: relaxed consistency of index information, used to rebuild index after failures
- Arbitrary levels of RLI hierarchy

A Flexible RLS Framework

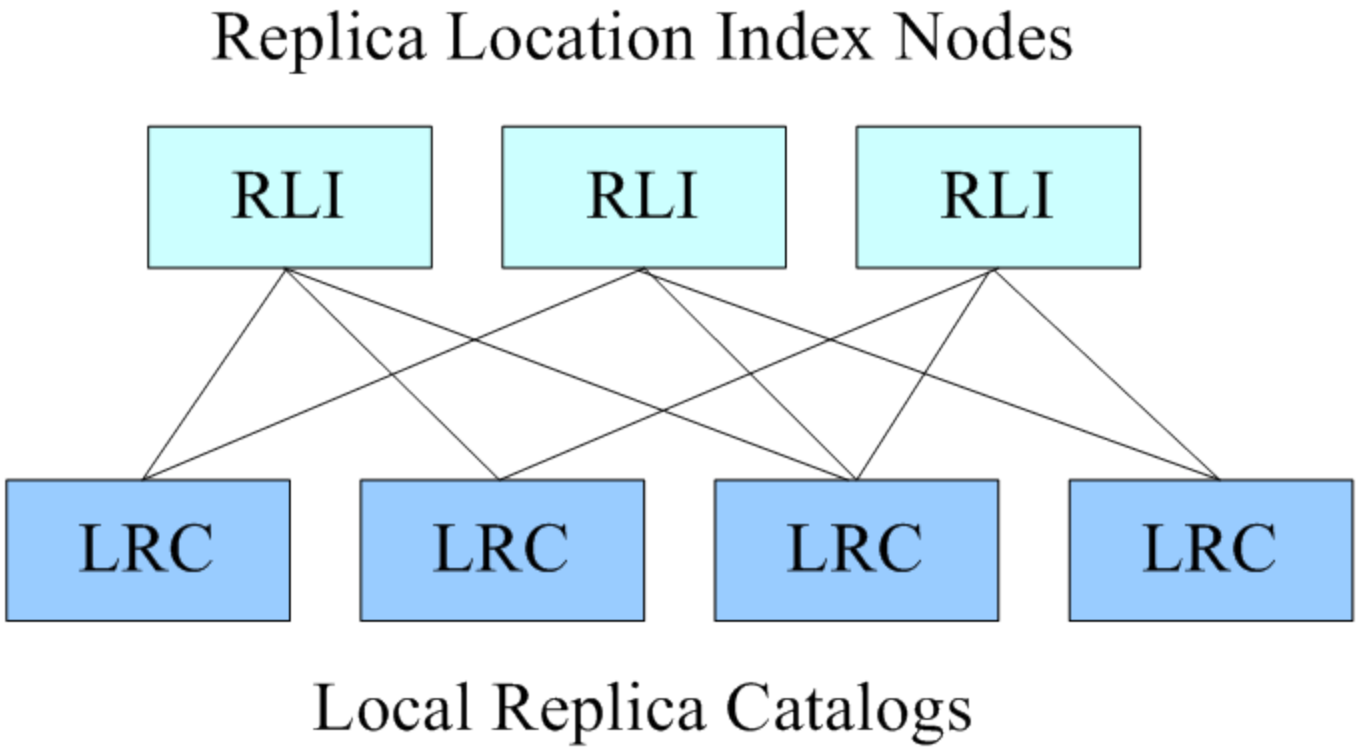
Five elements:

1. Consistent Local State: Records mappings between logical names and target names and answers queries
2. Global State with relaxed consistency: Global index supports discovery of replicas at multiple sites; relaxed consistency
3. Soft state mechanisms for maintaining global state: LRCs send information about their mappings (state) to RLIs using soft state protocols
4. Compression of state updates (optional): reduce communication and storage overheads
5. Membership service: for location of participating LRCs and RLIs and dealing with changes in membership

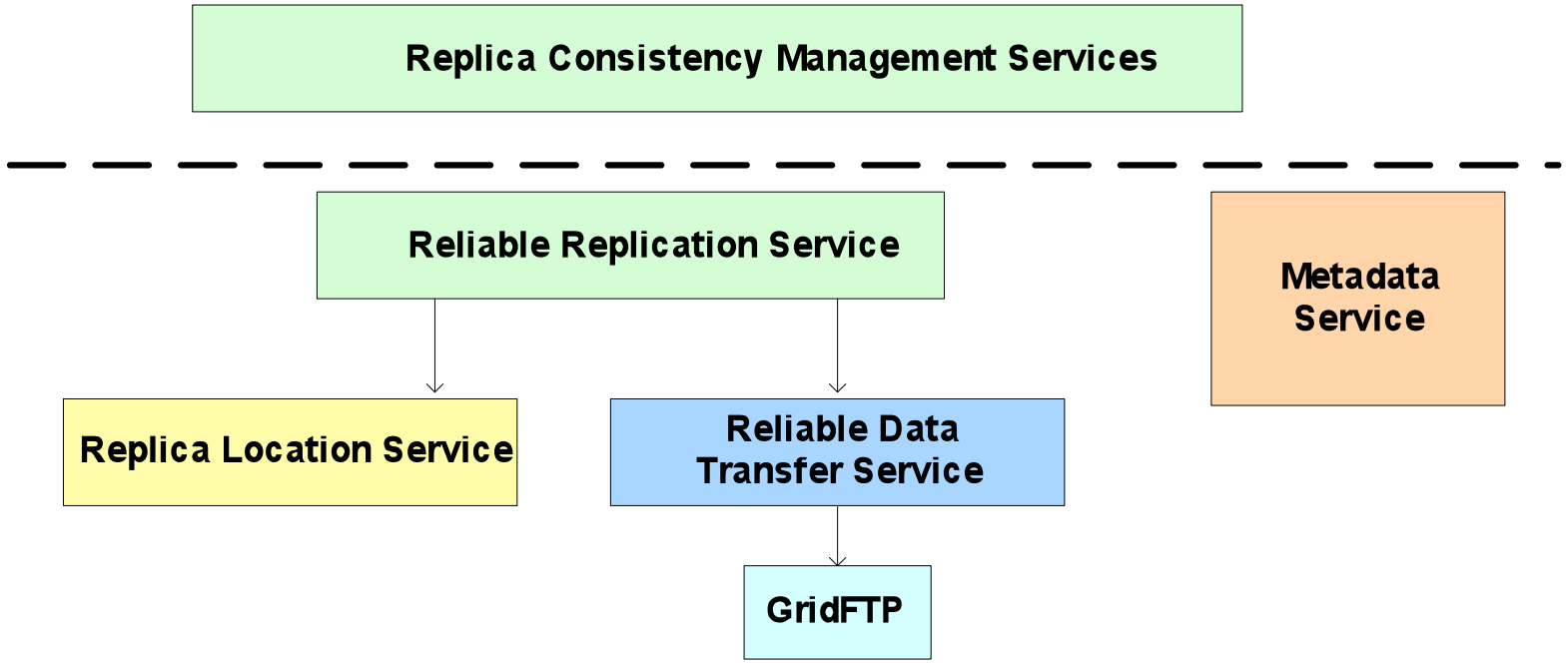
An RLS with No Redundancy, Partitioning of Index by Storage Sites



An RLS with Redundancy



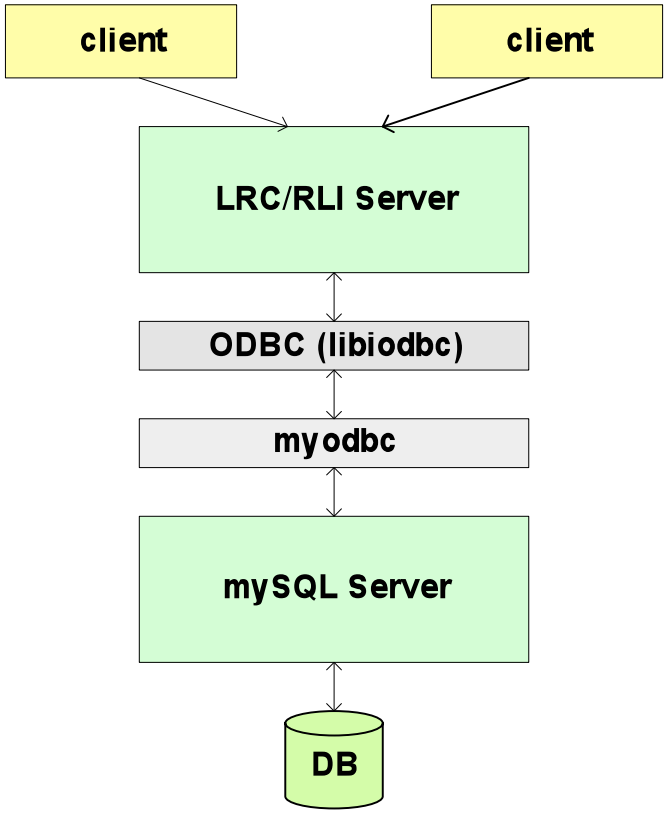
Replica Location Service In Context



- The Replica Location Service is one component in a layered data management architecture
- Provides a simple, distributed registry of mappings
- Consistency management provided by higher-level services

Components of RLS Implementation

- **Front-End Server**
  - Multi-threaded
  - Supports GSI Authentication
  - Common implementation for LRC and RLI
- **Back-end Server**
  - mySQL Relational Database
  - Holds logical name to target name mappings
- **Client APIs: C and Java**



Implementation Features

- Two types of soft state updates from LRCs to RLIs
  - Complete list of logical names registered in LRC
  - Bloom filter summaries of LRC
- User-defined attributes
  - May be associated with logical or target names
- Partitioning
  - Divide LRC soft state updates among RLI index nodes using pattern matching of logical names
- Membership service
  - Static configuration only
  - Eventually use OGSA registration techniques

Future Work

- Continued development of RLS
  - Code available as source and binary bundles at:
    www.globus.org/rls
    http://cern.ch/grid-data-management
- RLS will replace existing replica catalog API
- Reliable replication service being developed
  - Replicate data objects and register them in RLS
  - Provide fault tolerance
- RLS is part of the GT3 Alpha Release (as a GT2 service)
- RLS will become an OGSA grid service
  - Replica location grid service specification will be standardized through Global Grid Forum

RLS Installation

- Available as source and binary bundles
- Installation identical to other bundles except for setting some environment variables
- Example: to install the server binary bundle:

```
setenv GLOBUS_MYSQL_PATH /example/mysql
setenv GLOBUS_IODBC_PATH /example/iodbc
setenv ODBCINI /example/odbc.ini
gpt-install globus_rls_server-1.1-i686-pc-linux-gnu-bin.tar.gz
gpt-postinstall
```

Example application: Create new mapping (without error handling)

```
main(int argc, char **argv)
{
    globus_rls_handle_t *h;
    char error[MAXERRMSG];

    globus_module_activate(GLOBUS_COMMON_MODULE);
        globus_module_activate(GLOBUS_IO_MODULE);


    globus_module_activate(GLOBUS_RLS_CLIENT_MODULE);

    globus_rls_client_connect("rls://myserver", &h, error,
    MAXERRMSG);
        globus_rls_client_lrc_create(h, "lfn", "pfn");
        globus_rls_client_close(h);

    exit(0);
}
```

Example RLS Query

- In the former example, replace rls_client_lrc_create() with:

```
globus_list_t *list, *p;
globus_rls_string2_t *res;

globus_rls_client_lrc_get_lfn(h, "pfn", 0, 0, &list);

for (p = list; p; p = globus_list_result(p)) {
                res = (globus_rls_string2_t *) globus_list_first(p);
                printf("lfn %s: pfn %s\n", res->s1, res->s2);
}

globus_rls_client_free_list(list);
```